FPGAで学ぶVerilog HDL



第2回 LEDを点滅させる

デバイスの記事　ビギナーズ

井倉将実

前回（本誌2005年7月号，pp.113-119）の記事では，LSI設計技術習得の道具としてFPGA（field programmable gate array）を活用する効果を説明しました．また，最初のHDL（hardware description language）記述として，スイッチを押しているときにLEDが光り，離したら消えるという動作の回路を紹介しました．今回はまず，スイッチを押すたびにLEDの点灯/消灯が切り替わる回路を設計します．次に，1秒ごとに自動的にLEDの点灯/消灯が切り替わる（点滅する）回路を設計します．（筆者）

まず，前回のコードを少しだけ発展させて，スイッチを押すたびにLEDの点灯/消灯が切り替わる回路を設計します（**図1**）．LEDが点灯しているときにスイッチを押すとLEDは消灯し，LEDが消灯しているときにスイッチを押すとLEDは点灯するという動作を実現します．

表1　スイッチを押すたびにLEDの点灯/消灯を切り替える回路の動作

スイッチの状態	入力レベル	LEDの状態
押していない状態	“H”	変化なし
押した瞬間	“H”→“L”	直前まで点灯していれば消灯 直前まで消灯であれば点灯
押し続けた状態	“L”	変化なし
離した瞬間	“L”→“H”	変化なし

1.スイッチを押すたびにLEDの点灯/消灯を切り替える

前回，本連載で使うFPGAボード[(1)]でLEDを点灯させるには，LEDへの出力信号を“L”にすればよいと説明しました．そうなると，今回設計する回路の動作は，スイッチを押すごとにLEDに対して出力を“L”→“H”→“L”→…と変化させればよいことになります．

LEDの点灯と消灯を切り替えるのは，スイッチを押した瞬間とします．スイッチを離した瞬間や，スイッチを押し続けたとき，スイッチを離した状態では，何も変化させません．これをまとめると**表1**のようになります．

電気的に見ると，スイッチを押すことでFPGAのピンに“L”レベルの信号が入力されます．また，スイッチを離した（押していない状態）状態では“H”レベルです．

ここまでは前回と同じです．今回はもう少し細かく信号の変化を考えてみましょう．すると，スイッチを押した瞬間が存在することがわかります．このとき，信号は“H”から“L”に変化します．この変化が起こったときにLEDの点灯/消灯状態を逆の状態に切り替えれば，スイッチを押すたびにLEDの点灯/消灯を切り替える動作を実現できます．

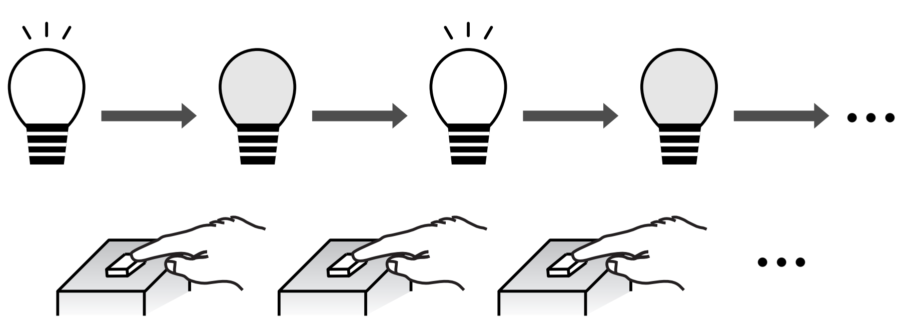


図1
スイッチを押すたびにLEDの点灯/消灯を切り替える
LEDが点灯しているときにスイッチを押すとLEDは消灯する．LEDが消灯しているときにスイッチを押すとLEDは点灯する．

リスト1　スイッチを押すたびにLEDの点灯/消灯を切り替える回路のVerilog HDL記述

```
module LED_DISP(
    SW_in ,
    LED_out
) ;

//---------------------------------------------------
// ポート宣言
//---------------------------------------------------
    input    SW_in ;
    output   LED_out ;

    reg      toggle_flag ;    // レジスタ型信号の宣言
```

```
//---------------------------------------------------
// 動作記述
//---------------------------------------------------
always @(negedge SW_in)
begin
    toggle_flag <= ~toggle_flag ;  // toggle_flagを反転する
end

// LED出力

    assign     LED_out  = toggle_flag ;
           // LED出力ピンにtoggle_flagを代入する

endmodule
```

```
always @（信号名） begin 動作 end
```

図2　always文
かっこの中に書かれた信号に変化があったときのみ，begin ～ end内の動作を行う．

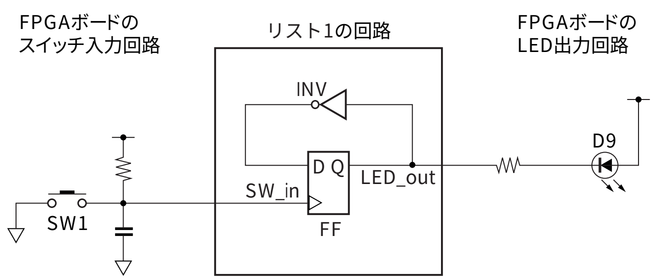


図3　スイッチを押すたびにLEDの点灯/消灯を切り替える回路のイメージ
フリップフロップと反転素子（インバータ）で構成される回路が生成される．

リスト1にスイッチを押すたびにLEDの点灯/消灯を切り替える回路のVerilog HDL記述を示します．

モジュール全体の構造は，前回示したコードと同じであることがわかります．前回と同じスイッチとLEDを利用するので，モジュール宣言とポート宣言も同じです．

● レジスタ宣言

リスト1では，内部信号toggle_flagをregで宣言しています．これは「レジスタ型」の宣言です．

レジスタは，記憶を行うモノの総称です．特定の入力信号に変化があったときにだけ出力を変化し，それ以外のときは入力信号が変化しても出力の状態を保持し続けます．

前回使用したwireによるネット型の信号は，状態を保持することはできません．入力の変化が直ちに出力に伝わる信号であり，assign文の左辺で使われます．逆に，レジスタ型で宣言した信号は，assign文の左辺では使用できません．

● always文

レジスタ型で宣言するだけでは，出力を保持することはできません．レジスタ型の信号は，always文といっしょに使うことで，状態を保持する回路になります．

always文の構文を**図2**に示します．かっこの中に書かれた信号に変化があったときのみ，begin ～ end内の動作を行います．

リスト1では，@(negedge SW_in)と記述してあります．これは，SW_in信号の立ち下がりエッジ，すなわち“H”から“L”への変化があったときを示しています．SW_in信号の立ち下がりエッジを認識した場合，begin ～ end内の動作を行うことになります．

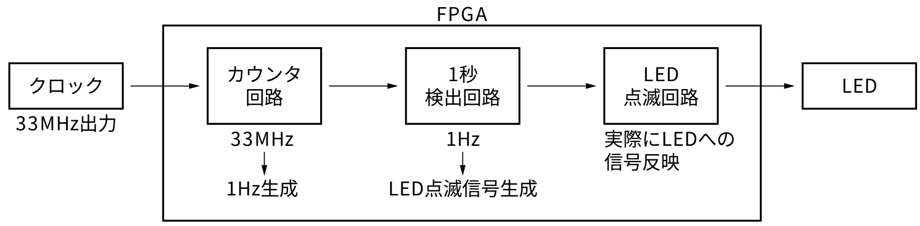
begin ～ end内は，レジスタtoggle_flagの論理を反転するという動作になります．「~」はNOT（反転論理）を表します．

結果として，**リスト1**のalways文で記述された回路の動作は，「SW_in入力の立ち下がりエッジを検出したらtoggle_flagの論理を反転する．SW_inの立ち下がりエッジ以外では，toggle_flagは状態を保持する」となります．

なお，negedgeが立ち下がりエッジを意味するのに対して，立ち上がりエッジ（“L”から“H”への変化）を表すときはposedgeを使います．

レジスタtoggle_flagは，そのままLED_outに接続します．LED_outはポート宣言された信号でFPGAの出力ピンになります．**リスト1**の記述では，暗黙のうちにネット型になります．assign文でtoggle_flagをLED_outに接続します．

図4
1秒ごとに自動的にLEDの点灯/消灯が切り替わる回路の構造
3,300万回を数えることができれば1秒のタイミングを作り出せる．これを使ってLEDを点滅させる．



● 動作確認

前回同様，CQ出版の「FPGAボードで学ぶ論理回路設計」に付属するFPGAボードを使って動作確認してみます．米国Altera社の設計ツールであるQuartus IIを使って，論理合成と配置配線を行います．ピン配置も前回と同じで，`SW_in`が34番ピン，`LED_out`が30番ピンです．

でき上がった回路のイメージを**図3**に示します．FPGAの内部に構成される回路も示しています．

リスト1の記述から，フリップフロップ（FF）と反転素子（インバータ）で構成される回路が生成されます．回路図による論理回路設計の経験がある方なら，動作をすぐに理解できるでしょう．

回路図になじみがない方には，少しわかりにくいかもしれません．FFは，信号入力とクロック入力，信号出力がある素子です．クロック入力が変化したとき，入力された信号が出力に現れます．そして，クロック入力が再び変化するまで，出力信号の状態を保ちます．そこで，「ある状態を保持する」機能を設計するときにFFが使われます．本稿は回路図による設計を趣旨としていないので，詳しい説明は教科書に譲りますが，ディジタルLSI設計では非常に重要な役割を持つ素子ですので，動作を覚えておいてください．

2.自動的にLEDを点滅させる

今まで紹介した回路は，すべて人間の手でスイッチ操作を行ってLEDを点滅させていました．次に，1秒ごとに自動的にLEDの点灯/消灯が切り替わる回路を設計します．

例えば時計であれば，人間がスイッチ操作を行わなくても動作を続けます．自動的に動作を行う回路を設計したい場合には，スイッチ以外の信号を使って回路を動作させることになります．ここで使用するのが「クロック」です．

● 動作の基本になるクロック

クロック信号を実際に発生するのは，「クロック発振器」という電子部品です．最近のディジタル機器であれば，ほとんどすべての回路で使われています．パソコンのCPUで「周波数は○○GHz」と言っているのもクロックのことです．多くのFPGAボードに，あらかじめ搭載されているはずです．そこで，本稿では「クロックはあらかじめ用意されている」として説明を進めます．

クロックは，“H”と“L”の変化をひたすら繰り返す信号です．本稿で使用しているFPGAボードの場合，33MHzのクロックが搭載されています．これにより，1秒間に3,300万回も信号の変化（“L”→“H”→“L”）が繰り返されます．このクロックを使うことで，LEDの点滅を自動的に行わせることができます．

とはいえ，人間の目でLEDが点滅していることを見分けられるのは，せいぜい1秒間に100回程度まででしょう．3,300万回もの点滅は認識できません．

● クロックから人間が識別できる周期の信号を作る

そこでまず，人間が識別できる周期で変化する信号をクロックから作り出し，そのタイミングでLEDを点滅させることにします．

1秒間に3,300万回信号の変化が繰り返されることがわかっているので，3,300万回を数えることができれば，1秒のタイミングを知ることができます．そこで，**図4**のようなしくみで回路を構成します．

リスト2に1秒ごとにLEDの点滅を繰り返す回路のVerilog HDL記述を示します．**リスト1**と比べて，かなり複雑に見えるかもしれません．しかし，新しく必要になる知識はそれほど多くありませんので，一つ一つじっくりと理解してください．

ここでは，入力信号名をCLKに変更しました．スイッチではなくクロックを使うことを明示するためです．

● パラメータ宣言

`parameter`で記述されている部分は，パラメータ宣言

リスト2　1秒ごとに自動的にLEDの点灯/消灯が切り替わる回路のVerilog HDL記述

```
module LEDLIGHT(
     CLK,
     LED_OUT
) ;

/****************************************/
     input      CLK ;
     output     LED_out ;

     wire       LED_out ;

/****************************************/
     parameter  F33M000_cnt= 32’h01F78A40 ;
                          // 33.000MHz時の1秒設定

     reg        [31:0]    sec_cnt ;// 1秒作成用カウンタ
     reg        sec1_flag ;    // 1秒を示すフラグ

     reg        toggle_flag ;// 1秒ごとにトグルするフラグ

/****************************************/
/*   1秒カウンタ部                        */
/****************************************/

always @(posedge CLK)
begin
     if ( sec_cnt == F33M000_cnt ) begin  // もし3300万回数え
       sec_cnt  <= 32’h00000000 ; //  たらカウンタをリセットする
       sec1_flag  <= 1’b1 ;//  SEC1_FLAGに1をセットする
     end else begin              //  それ以外だったら
       sec_cnt  <= sec_cnt + 1 ; //  カウンタは＋1ずつ数え
る
       sec1_flag <= 1’b0 ; //  SEC1_FLAGを0にリセットする
     end
end

/****************************************/
/*   1秒検出部. LED出力準備               */
/****************************************/

always @(posedge CLK)
begin
     if ( sec1_flag == 1’b1 ) begin
       toggle_flag <= ~toggle_flag ;
     end
end

/****************************************/
/*  LED出力部                     */
/****************************************/

     assign    LED_out   = toggle_flag ;
                   // LED出力. toggle_flagを使う.

/****************************************/
```

です．ある名まえの変数に対して数値を割り当てるときに使います．記述を見やすくし，ミスをなくす効果があります．例えば，タイミングやバス幅，1秒のような基準となる値は，回路のさまざまな箇所で使われます．そのような定数に変更があったとき，すべてを書き直すのはたいへんですが，parameterで定義しておけば1ヵ所の修正ですみます．

パラメータ宣言の構文を**図5**に示します．例えば今回は3,300万回を数えたいので，F33M000_cnt[注1]という名まえに32'h01F78A40を与えています．32'h01F78A40という表現の意味は，「32 '」が32ビット幅，「h」が16進数，「01F78A40」が16進数による3,300万の値です．

今回使用しているボードは，製造時期によってクロック周波数が異なります．具体的には，32MHz，33MHz，33.333 MHzの3種類があります．そこでparameterでパターンをあらかじめ用意し，周波数に応じて切り替えやすくしています．32MHzなら32'h01E84800，33.333 MHzなら32'h01FCA055に変更するだけです．

● ポート宣言

ポート宣言では，三つのレジスタ型信号を宣言しています．これらのうち，sec_cntには[31:0]という記述が付いています．これは32ビット幅のバス信号であることを示しています．

```
parameter  変数の名まえ ＝ 数値 ;
```

図5　パラメータ宣言
変数に対して数値を割り当てる．

32ビット幅のバス信号sec_cntは，3,300万回を数えるために使います．sec1_flagは1秒ごとに1クロックだけ有効になる信号です．toggle_flagは，1秒間隔で‘1’と‘0’の変化を繰り返す信号です．

● カウンタ回路

最初のalways文は，33MHzのクロックから1秒のタイミングを得るための記述です．

32ビット幅のバス信号sec_cntを0から，クロックの立ち上がりエッジごとに＋1ずつ加算していくと，sec_cntが3,300万（32'h01F78A40）になったときがちょうど1秒ということになります．そこで，sec_cntが3,300万になったら，sec1_flagを‘1’にします．同時に，sec_cntは再び0からカウントを始めます． sec_cntが3,300万以外のときは，sec1_flagを‘0’とします．

CLK信号の変化で内部の回路を動かすので，always文

注1：「F33M000_cnt」は，F（frequency；周波数），33.000MHz，_cnt（カウント値）という意味であり，筆者の命名規則によるもの．

図6
if文
条件によって異なる動作を行うときに使う.

```
if （ 条件式 ） begin
      条件が一致したときの動作
end else begin
      条件が不一致のときの動作
end
```

のかっこ内にはCLKを入れます．CLKの立ち上がりエッジごとに動作させるために，posedgeを使います．

● if文

ここでは，

- sec_cntが3,300万回数えたら，sec_cnt信号をリセットし，sec1_flagを‘1’(1'b1)にセットする．
- それ以外はsec_cnt信号は＋1ずつ数え続け，sec1_flagを‘0’(1'b0)にする．

というような2種類の動作をさせる必要があります．

条件によって異なる動作をさせたい場合，if文を使います．if文の構文を**図6**に示します．

if文の条件判断の部分で，3,300万を示す値を直接記述するのではなく，パラメータ宣言したF33M000_cntを使っているところがポイントです．

また，sec_cnt + 1というのが，sec_cntを1だけ加算する記述です．Verilog HDLでは，＋や－を使った算術演算が可能です．

if文はかならずalways文の中で使わなければなりません．このとき，条件判断を行うタイミングをalwaysのかっこ内に記述する必要があります．

● 1秒ごとのタイミングの検出

sec1_flagは，3,300万回を数えたときだけ‘1’になり，それ以外は‘0’に戻されます．つまりsec1_flagフラグ信号は，3,300万分の1秒間しか‘1’になりません．この信号を使って，LEDを点灯/消灯することを考えます．

sec1_flagは3,300万分の1秒しか‘1’ではないのですが，‘1’になってから次の‘1’が出現するまでの時間は1秒になっています．

そこで，CLK信号の立ち上がりエッジごとにsec1_flagの状態を検出して，もし‘1’であればLEDへの出力信号を反転させればよいことになります．

この動作をalways文とif文で記述しています．ここで使用している記述は，すべてすでに説明ずみです．難しくありませんので，よく考えてみてください．

LEDへの出力を行うassignについても，これまでと同じです．

● 動作確認

Altera社のQuartus IIを使って，論理合成と配置配線を行います．ピン配置は，CLKが39番ピン，LED_outが30番ピンです．

3.まとめ

クロックを使って，ある決まった一定のリズムで回路を動かすことができました．今回説明を行ったクロックとalways文を使った記述が，Verilog HDLによるLSI設計の基本です．

考えてみると，たしかに人間の動作でも，あるイベントが起きたときは直ちに処理しなければなりません．一方，イベントがない場合はそれまでの状態を保っていることが多いようです．上司から要求が来た場合，それまで続けていたしごとから新しいしごとに切り替わります．なにやら回路も同じように思えませんか．

今回の1秒カウンタはクロックの立ち上がりエッジごとに内部の回路を呼び出します．これはお客様から呼び出しが来るたびに筆者が返事をする，まさに「パブロフの犬」状態の回路ではないかと，今回執筆していて思いました．

1秒のタイミングを作り，LEDを点滅させることができました．これを発展させると時計ができそうだと思われた方も多いのではないでしょうか．次回は，時計を作るということを考えながら説明を進めることにします．

参考・引用*文献

(1) 山際伸一；FPGAボードで学ぶ論理回路設計，CQ出版社，2003年5月．

いくら・まさみ
FPGA-Magician

<筆者プロフィール>
井倉将実．今回はタイ，アユタヤの世界遺産を見つつ執筆しました．最新FPGA設計を使った案件で，開発はタイ，出荷先は米国，打ち合わせは日本と激務に輪をかけています．ことばの壁はありますが，Verilog HDLがあればお互いに理解しあえるので，なんとかなってます．